JOHANNES GUTENBERG UNIVERSITÄT MAINZ

NVIDIA AI Technology Center

WarpCore
A Library for fast Hash Tables on GPUs

Daniel Jünger[1], Robin Kobus[1], André Müller[1], Christian Hundt[2], Kai Xu[3], Weiguo Liu[3], Bertil Schmidt[1]

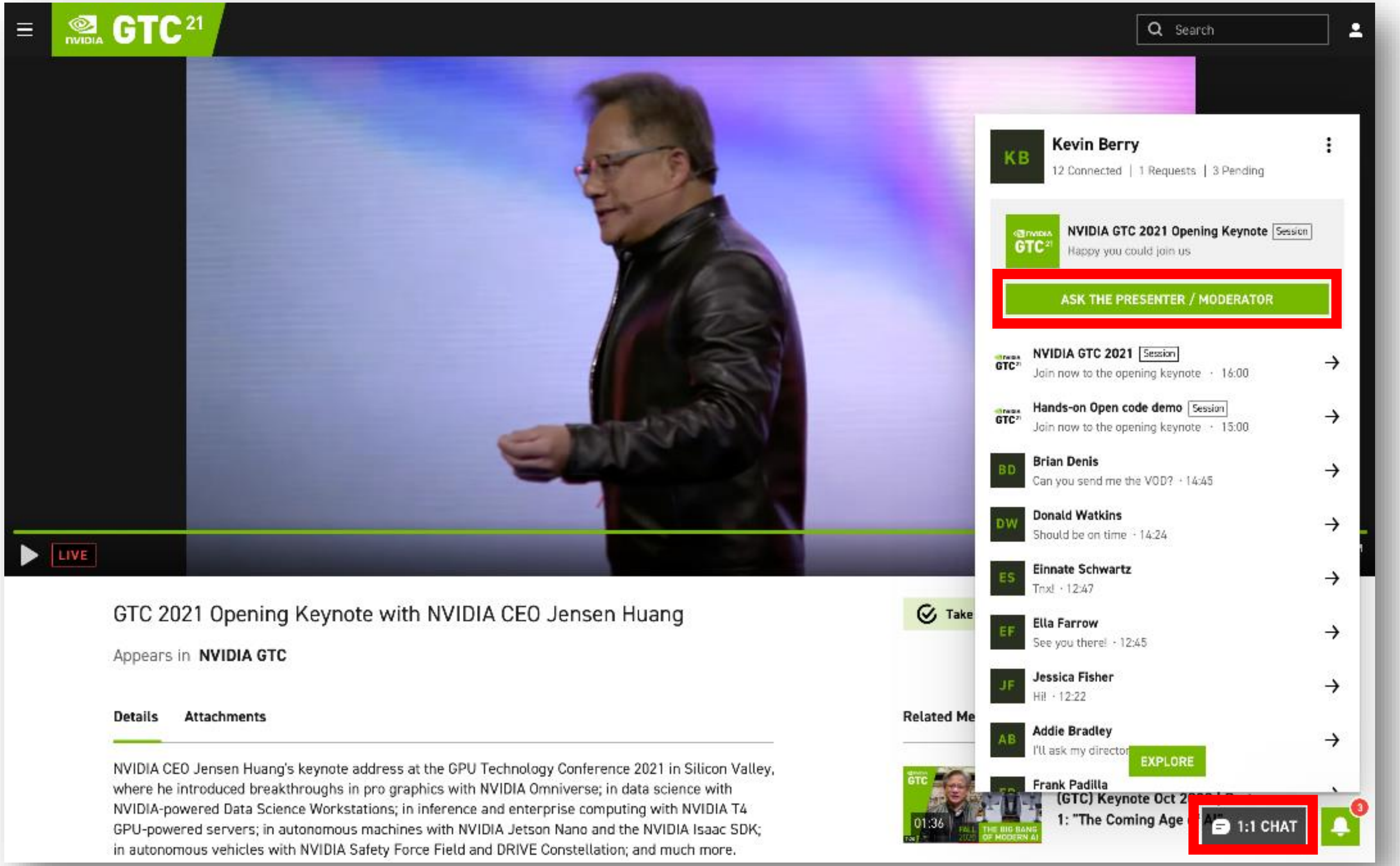
[1] Institute of Computer Science, Johannes Gutenberg-University, Mainz, Germany

[2] NVIDIA AI Technology Center Europe

[3] School of Software, Shandong University, Jinan, China

GPU Technology Conference 2021

WE ARE AVAILABLE TO CHAT DURING THIS SESSION



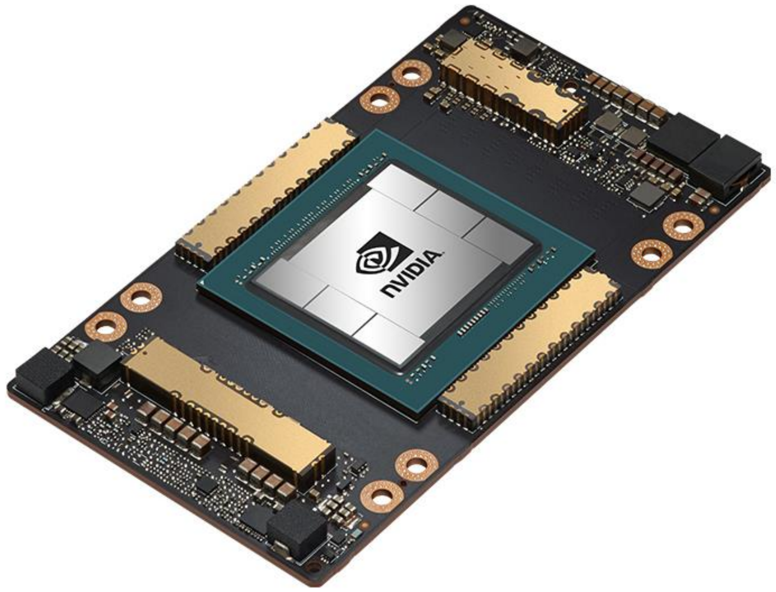
Click on “1:1 Chat,” then “Ask the Presenter / Moderator” button to submit your question.
After the session is over, connect with me via attendee chat by searching for my name.

IT IS ALL ABOUT MEMORY BANDWIDTH

RAM modules in CPU servers

a few hundreds of GB/s
a few TB of size

HBM2 stacked memory modules attached to a Tesla A100



up to 2 TB/s
less than 80 GB (A100)

WHY HASHING IS A GOOD IDEA

Hash tables are well-suited if range queries do not matter:

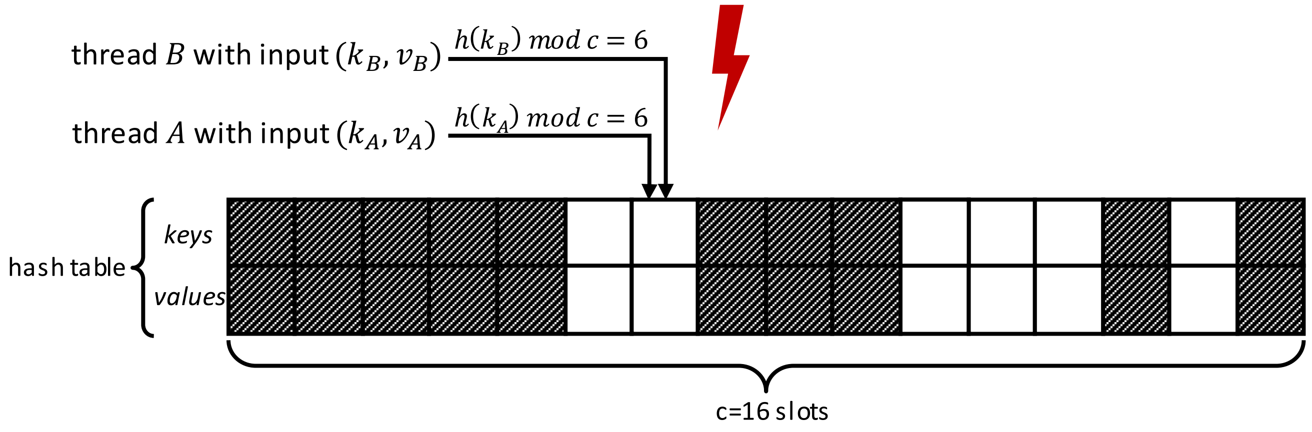
	Hash Table	Sorted Array	Tree
insertion per element	$O(1)$	$O(\log n)$	$O(\log n)$
query per element	$O(1)$	$O(\log n)$	$O(\log n)$
peak memory	$(1 + \varepsilon)n$	$2n$	$(1 + \varepsilon)n$
final memory	$(1 + \varepsilon)n$	n	$(1 + \varepsilon)n$
range queries	not supported	supported	supported

- out-of-place sorting usually needs $O(n)$ auxillary memory: CUDA Unbound radix sort uses double buffers → waste of valuable video memory
- incomplete trees exhibit highly irregular data layouts and are hard to construct in parallel without auxillary memory

PARALLEL HASH TABLE CONSTRUCTION

Scenario: inserting new key/value pairs into a hash table in parallel

- determine slot index for k_A by applying a hash function $h(k_A)\ mod\ c = 6$
- write (k_A, v_A) to the target slot
- subsequent retrieval of the same element works in the same fashion



- **hash collisions** among keys
  - $h(k)\ mod\ c = h(k')\ mod\ c$ for $k \neq k'$
  - for suitable resolution strategies see next slide
- **race conditions** in a parallel setup
  - can be avoided by using atomic operations (CAS)

COLLISION RESOLUTION STRATEGIES

Separate Chaining

Slots (buckets) store multiple colliding key-value pairs.

- **Dynamic Linked Lists**
  - allows for dynamic table growth
  - overhead due to memory allocations
  - slow pointer chasing during bucket iteration
- **Static Arrays**
  - memory over-provisioning
  - requires additional array iteration during probing

Open Addressing

Find the next unoccupied slot by means of a deterministic probing scheme.

- **Linear Probing:** $\boldsymbol{s(k,i) = (h(k) + i) \bmod m}$
  - cache efficient
  - prone to primary and secondary clustering
- **Quadratic Probing:** $\boldsymbol{s(k,i) = \left(h(k) + i^2\right) \bmod m}$
  - leaves dense regions faster than linear probing
  - prone to secondary clustering, i.e., $s(k,0) = s(k',0)$
- **Double Hashing:** $\boldsymbol{s(k,i) = \left(h_1(k) + i * h_2(k)\right) \bmod m}$
  - if m is prime and $0 < h_2(k) < m$ then
    - $s(k,0) \neq s(k',0)$, i.e., no secondary clustering
    - $s(k,i)$ for $i < m$ is cycle-free
- **Cuckoo Hashing**
  - greedily swap keys between candidate positions
  - may result in infinite cycles
- **Robinhood Hashing**
  - takes from the rich and gives to the poor
  - reduces probing length variance

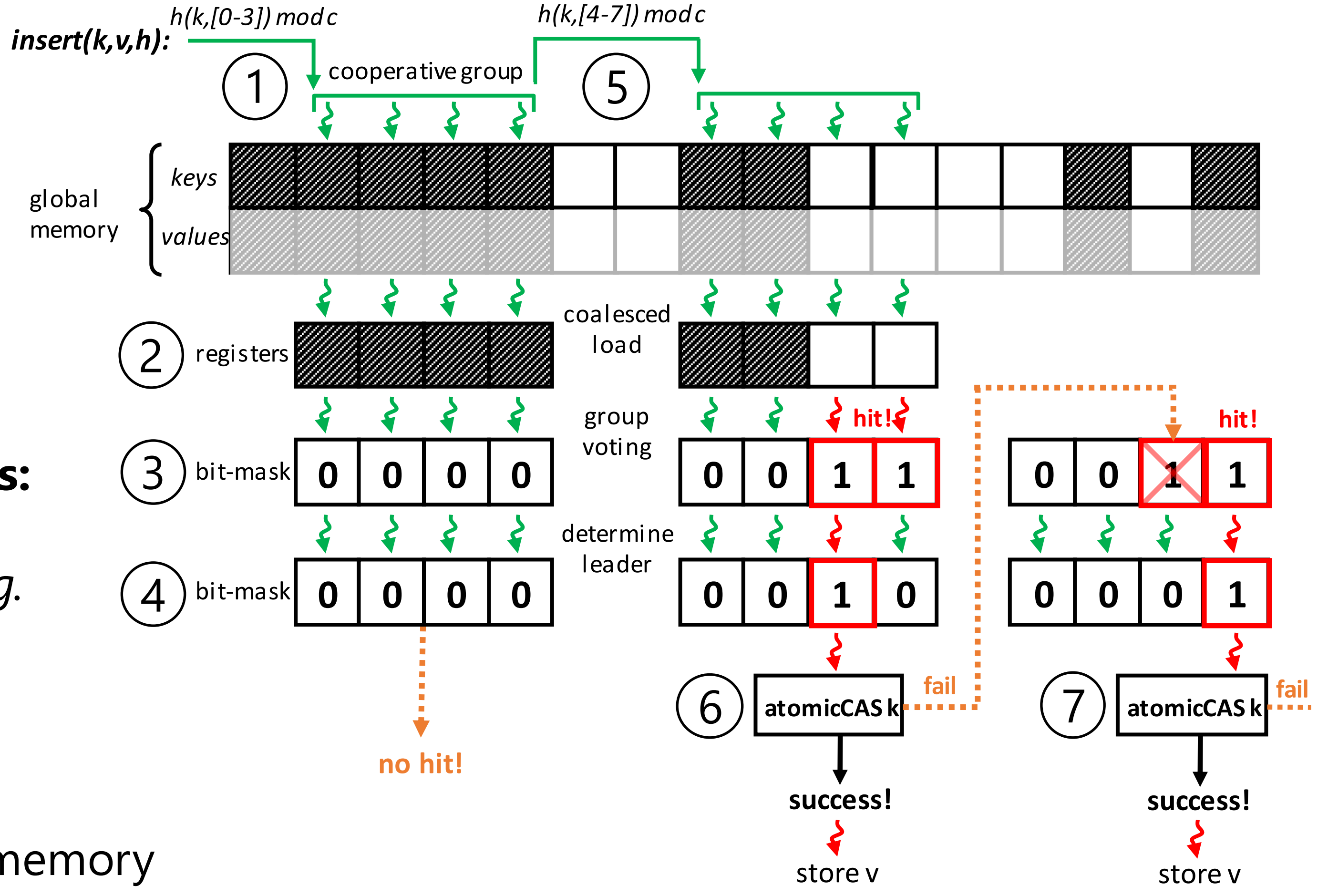
CONTRIBUTIONS

We propose WarpCore - a versatile library of hashing data structures

- **Performance**
  - main focus on high-throughput table operations
  - WarpCore outperforms other state-of-the-art CPU and GPU hash tables
- **Modularity**
  - building blocks for constructing customized GPU hash tables
  - probing schemes, hashers, memory layouts, etc.
- **Host-sided and device-sided interfaces**
  - host-sided (bulk) operations provide high throughput
  - device-sided operations (fuse table operations with other tasks in one kernel)
- **Fully-asynchronous execution**
  - allows for task overlapping and multi-GPU setups

COOPERATIVE PROBING SCHEME

- exploits fast intra-warp communication via registers
- intra-group linear probing
  + inter-group chaotic probing



Considerations for multi-value scenarios:

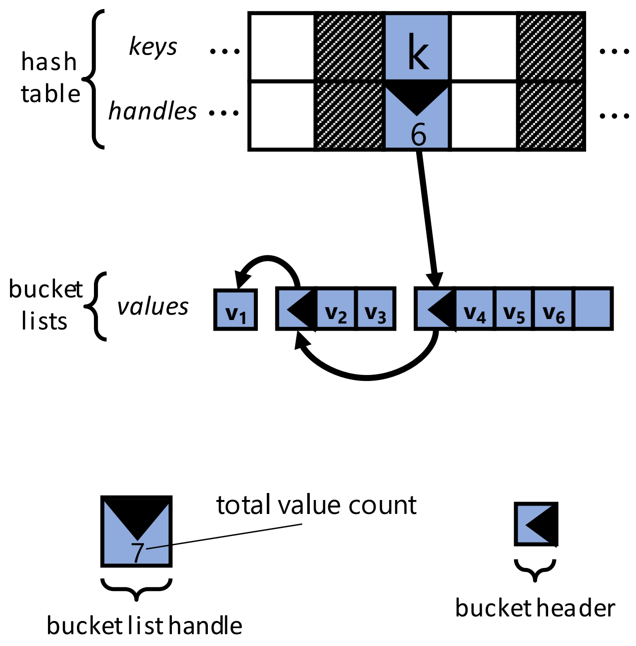
- probing scheme has to be *cycle-free (e.g. double hashing)*
- retrieval can be done cooperatively
- storing identical keys multiple times is memory inefficient

BUCKET LIST HASH TABLE

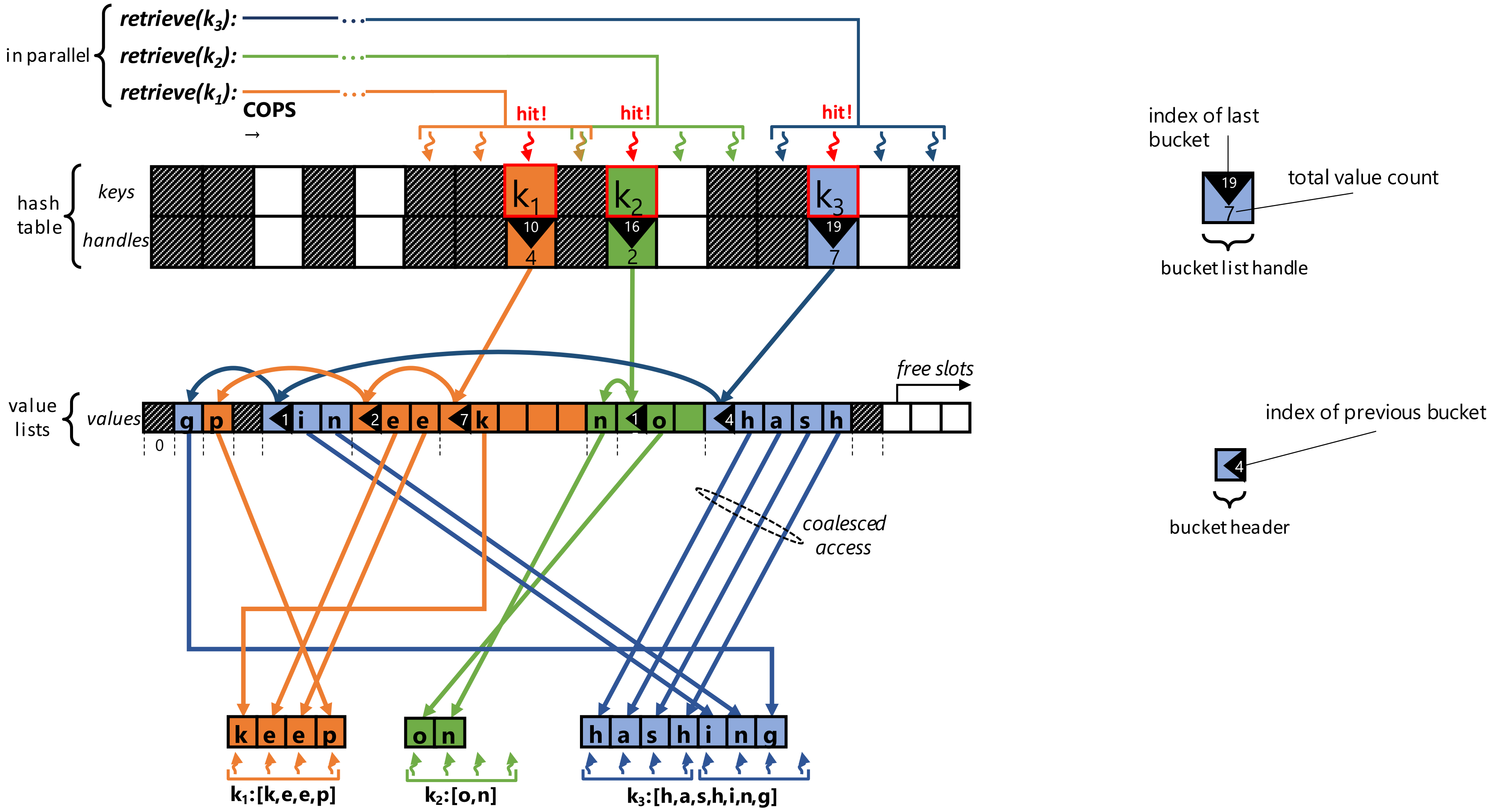
Open adressing hash tables lack space efficiency for highly skewed data.

Alternative approach:

- store keys only once in a single-value OA hash table
- each key holds a handle to a list of values
- each list consists of linked buckets of varying size
- buckets reside inside a pre-allocated memory pool

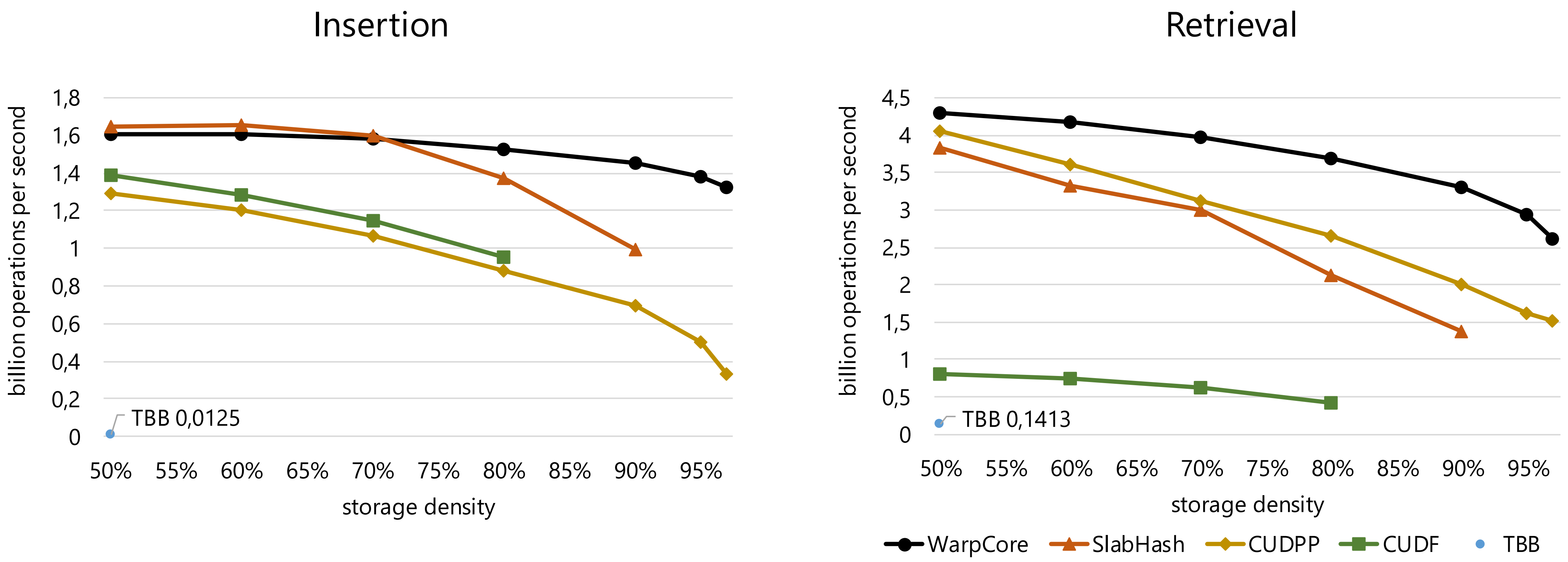


BUCKET LIST HASH TABLE



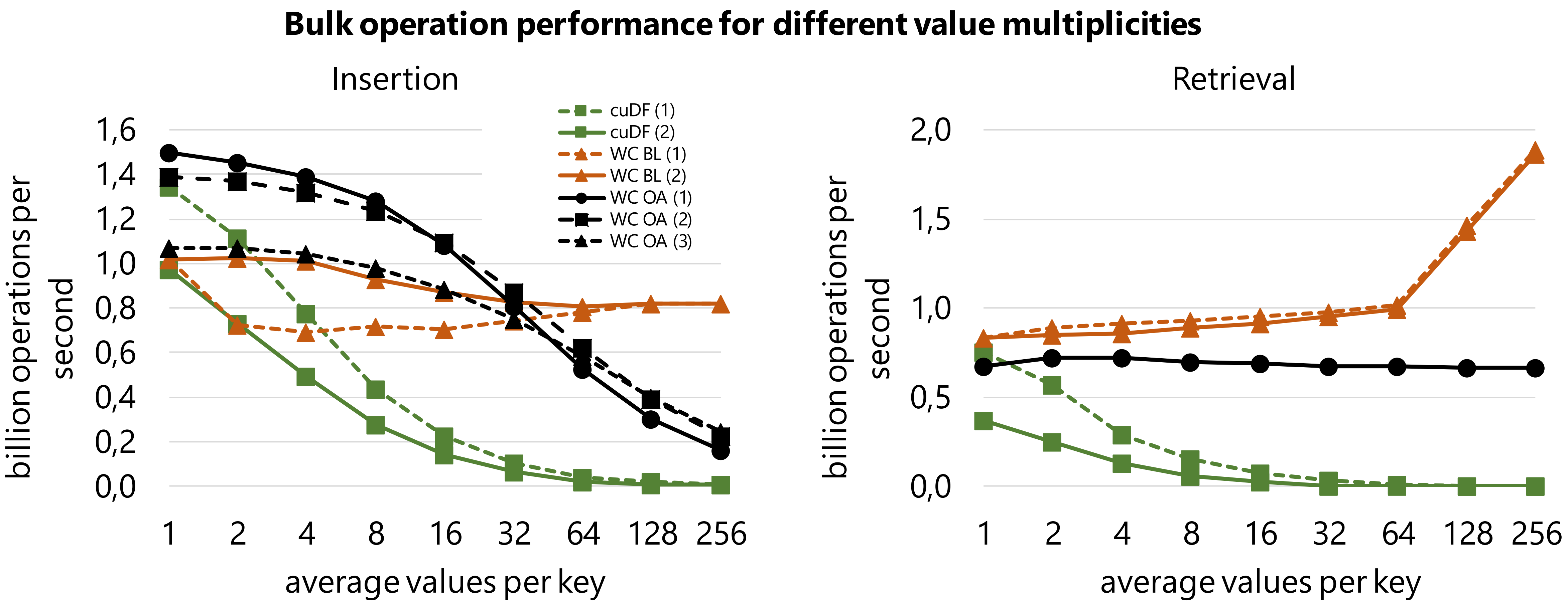
SINGLE-GPU SINGLE-VALUE PERFORMANCE

Bulk operation performance with 2^{28} (2 GB) unique (4 byte)key-(4 byte)value pairs.



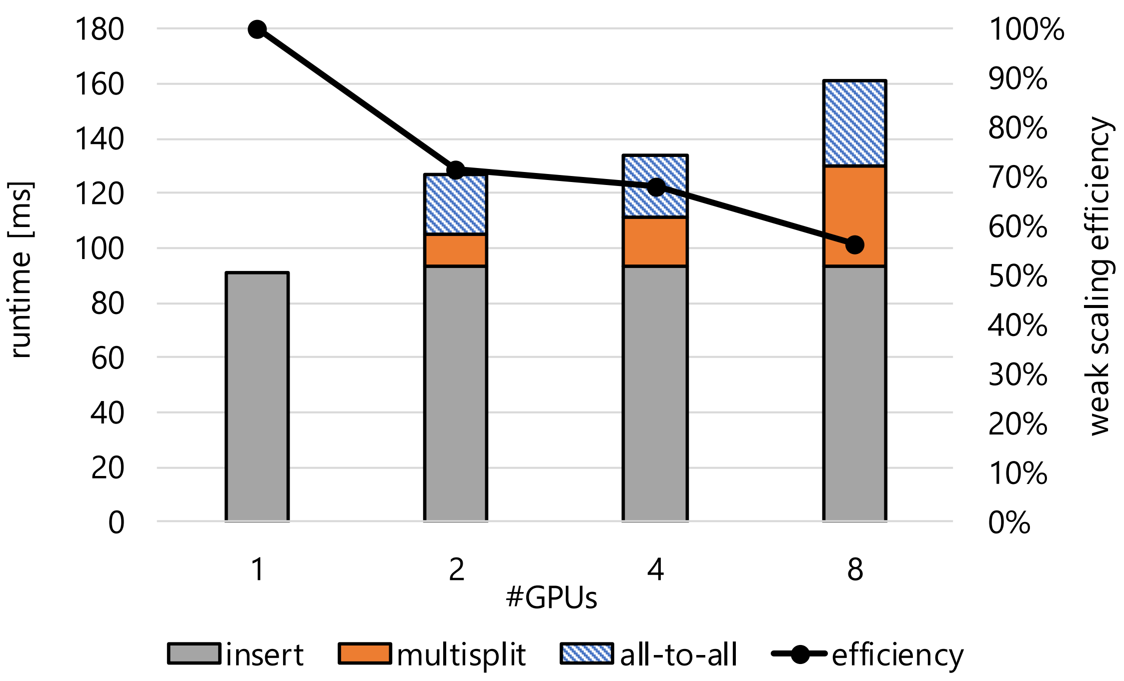
Benchmarks were conducted on a NVIDIA GV100 (Volta) GPU

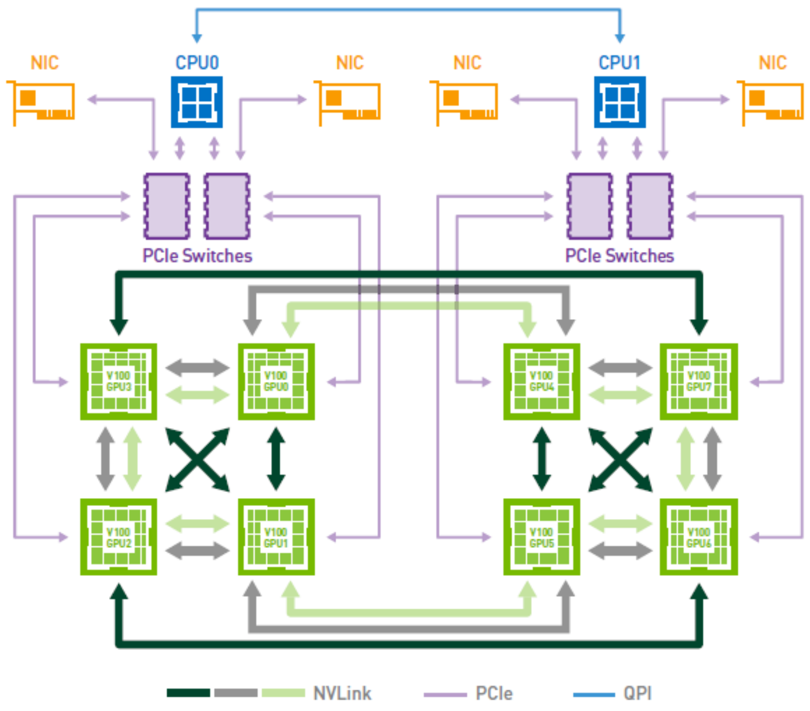
SINGLE-GPU MULTI-VALUE PERFORMANCE



Benchmarks were conducted on a NVIDIA GV100 (Volta) GPU

Weak scalability analysis on a DGX-1 server with 2GB of key-value pairs per GPU.

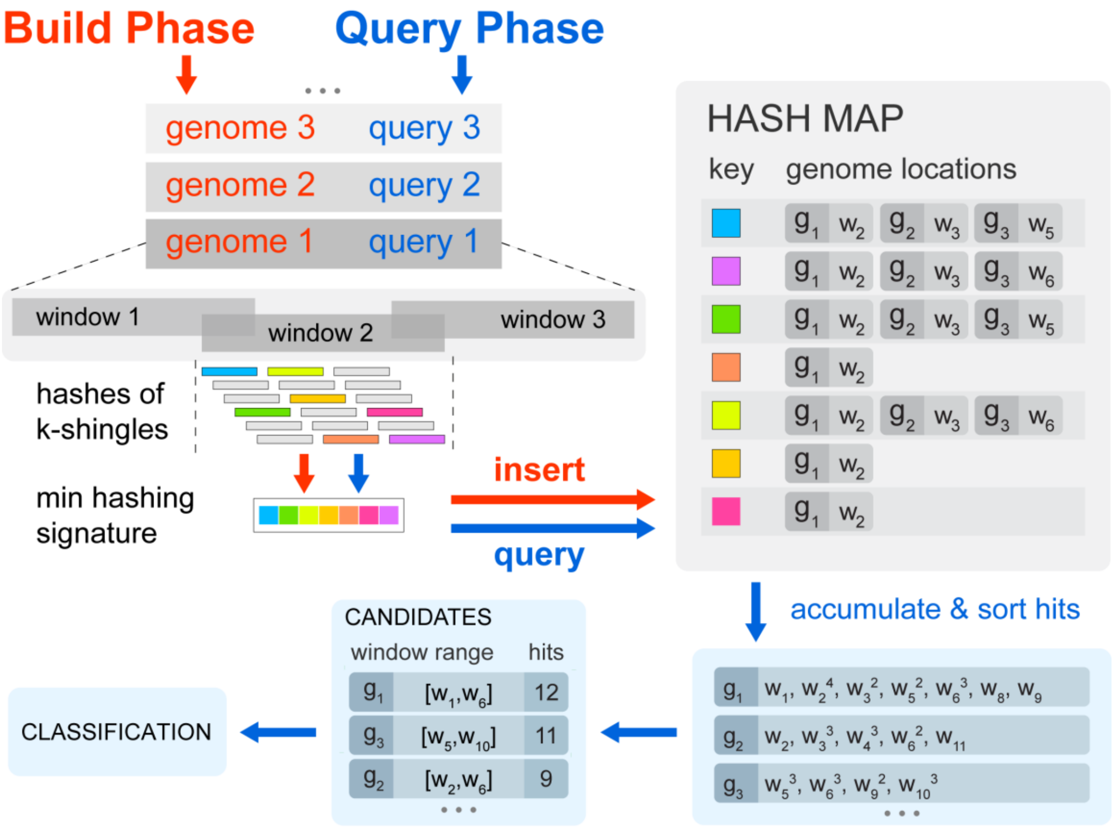


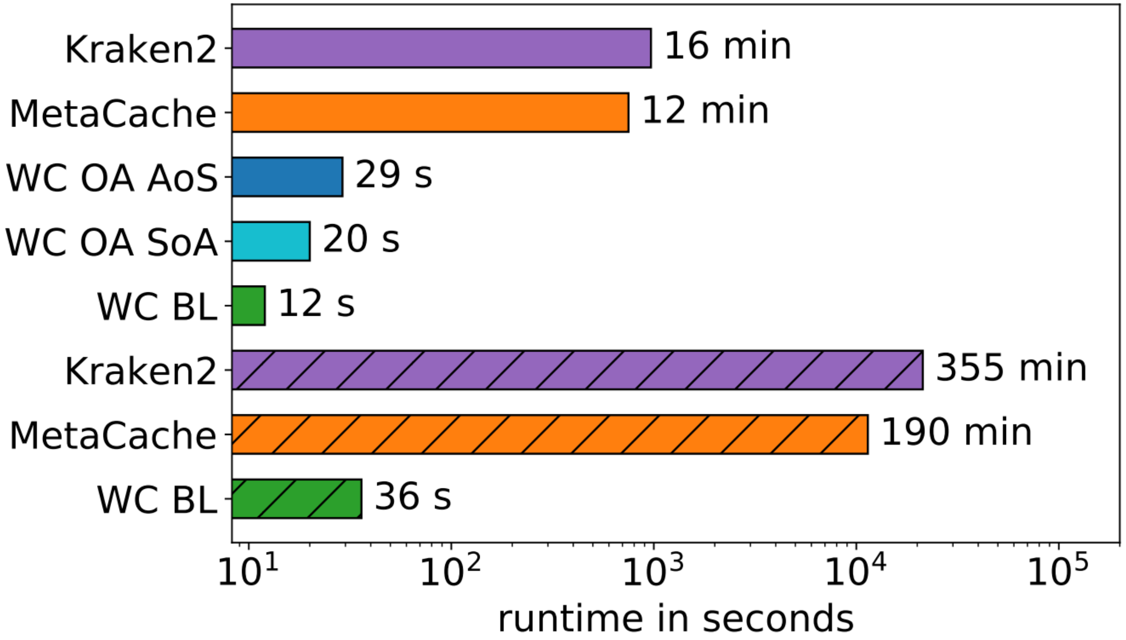


WarpCore achieves 100.8 GB/s throughput using 8 Tesla V100 at a scaling efficiency of 53%.

APPLICATION: METAGENOMIC CLASSIFICATION

Task: Determine taxonomic composition of an environment sample





solid bars: 18 GB database on one GV100
hatched bars: 120 GB database on 8 V100 (DGX-1)

CONCLUSIONS

We have presented WarpCore - a versatile library of GPU hash table data structures.

- a framework for high-throughput hashing-based data structures that can be tailored to fit many use cases
- efficient implementations of single- and multi-value hash tables, hash sets, counting hash tables, and bloom filters
- we propose a new multi-value hash table approach which provides robust throughput at high memory densities even for highly skewed input distributions
- easily scalable over up-to 16 GPUs (DGX-2) or a DGX-A100 (640 GB available GPU memory)

THANK YOU!

- Daniel Jünger, Robin Kobus, André Müller, Bertil Schmidt
  - {juenger, kobus, muellan, bertil.schmidt}@uni-mainz.de
  - Johannes Gutenberg University, Mainz, Germany
- Christian Hundt
  - chundt@nvidia.com
  - NVIDIA AI Technology Center
- Kai Xu, Weiguo Liu
  - {xukai16@mail., weiguo.liu@}sdu.edu.cn
  - School of Software, Shandong University, Jinan, China

- https://github.com/sleeepyjack/warpcore (Apache 2.0 License)
- [2009.07914] WarpCore: A Library for fast Hash Tables on GPUs (arxiv.org)

RELATED TALKS ON GTC 2021

- Fast and Simple Hash Tables [S31466]
  - Monday April 12th, 10am PDT
- Optimizing Data Movement with Compression for Database Applications [S31645]
  - Monday April 12th, 10am PDT
- Eliminating DRAM Random Access Bottleneck from GPU Hash Join Algorithm [S31373]
  - Monday April 12th, 10am PDT
- Multi-GPU HashGraph: A Scalable Hash Table for NUMA Systems [S31527]
  - Thursday April 15th, 10am PDT

Q&A

What is the worst-case distribution of keys for your hash map?

Q&A

What GPUs does WarpCore support?

Q&A

How can I get started with WarpCore?